中華民國經濟部智慧財產局

INTELLECTUAL PROPERTY OFFICE
MINISTRY OF ECONOMIC AFFAIRS
REPUBLIC OF CHINA

茲證明所附文件，係本局存檔中原申請案的副本，正確無訛，
其申請資料如下 ：

This is to certify that annexed is a true copy from the records of this office of the application as originally filed which is identified hereunder:

申　　請　　日 ： 西元 2002 年 12 月 27 日
Application Date

申　請　案　號 ： 091137721
Application No.

申　　請　　人 ： 財團法人工業技術研究院
Applicant(s)

局　　長
Director General

蔡練生

發文日期： 西元 2003 年 2 月 17 日
Issue Date

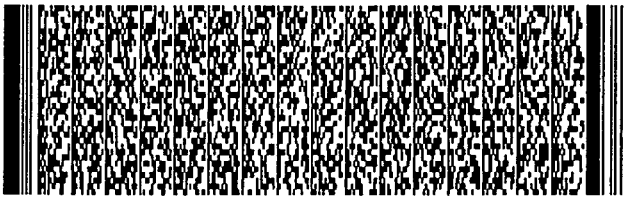
發文字號： 09220145110
Serial No.

申請日期：	案號：
類別：	

（以上各欄由本局填註）

發明專利說明書

一、發明名稱	中 文	保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法與裝置
	英 文	Method and Apparatus for Protecting Public Key Schemes from Timing, Power and Fault Attacks
二、發明人	姓 名（中文）	1. 顏嵩銘 2. 呂誌忠 3. 曾紹銎
	姓 名（英文）	1. Sung-Ming YEN 2. Chih-Chung LU 3. Shau-Yin TSENG
	國 籍	1. 中華民國 2. 中華民國 3. 中華民國
	住、居所	1. 桃園縣中壢市五權里38號 2. 臺北縣樹林市中山路二段48號2樓 3. 新竹縣峨眉鄉富興村8鄰13號
三、申請人	姓 名（名稱）（中文）	1. 財團法人工業技術研究院
	姓 名（名稱）（英文）	1. Industrial Technology Research Institute
	國 籍	1. 中華民國
	住、居所（事務所）	1. 台灣省新竹縣竹東鎮中興路四段195號
	代表人姓 名（中文）	1. 翁政義
	代表人姓 名（英文）	1. Cheng-I WENG



四、中文發明摘要　(發明之名稱：保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法與裝置)

本發明提供一保護公開金鑰系統以防止時間，電力與錯誤攻擊之方法，有別於傳統無法防止時間攻擊和簡單電力攻擊的演算法。本發明提供了一指模演算法，其中不包含條件指令以防止時間攻擊及簡單電力攻擊；並且演算法中沒有多餘的運算，使得攻擊者無法由藉由產生C safe error來得知秘密金鑰的資訊，且演算法中不包含被乘數乘以乘數存回不固定位址之暫時儲存體的操作，使得M safe error的攻擊不會對秘密金鑰產生威脅。

代表圖：第五圖；
元件符號：100-150 流程步驟。

英文發明摘要　(發明之名稱：Method and Apparatus for Protecting Public Key Schemes from Timing, Power and Fault Attacks)

The present invention provides a method for protecting public key schemes from timing, power and fault attacks. In general, this is accomplished by implementing critical operations using "branchless" or fixed execution path routines whereby the execution path does not vary in any manner that can reveal new information about the secret key during subsequent operations. More particularly, the present invention provides a modular exponentiation algorithm without any

四、中文發明摘要　(發明之名稱：保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法與裝置)

英文發明摘要　(發明之名稱：Method and Apparatus for Protecting Public Key Schemes from Timing, Power and Fault Attacks)

redundant computation so that it can protect the secret key from C safe error attacks. The improved method also provides an algorithm that doesn't have a store operation with non-certain destination so that the secret key is immune from M safe error attacks.

本案已向

國(地區)申請專利	申請日期	案號	主張優先權
無			

有關微生物已寄存於	寄存日期	寄存號碼
無		

五、發明說明（1）

一、【發明所屬之技術領域】

本發明係為一種指模運算演算法，此法不但能抵抗時間攻擊技術與電力攻擊技術的攻擊，並且能抵抗錯誤攻擊技術的攻擊。

二、【先前技術】

在傳統密碼系統裡，密碼演算法可以被轉成數學模型，藉由分析數學模型，使大家相信要找出低複雜度的演算法來破解密碼演算法是不可能的。然而這些數學模型分析並沒有考慮密碼演算法實作的問題。由於在密碼演算法的實作過程中，都有一些邊際通道沒有被包含在其數學模型中，這允許攻擊者經由非正式管道破解所謂安全的密碼演算法。非正式管道包含探針刺探、反向工程、儲存體讀取技術等，這些必須依賴專業的知識與儀器設備才可辦到。為了防止這些攻擊，當今智慧卡會作下列的基本防備措施，如在其晶片塗上一層保護層以做保護，在晶片的最上層做一層網狀的金屬層，智慧卡內含時脈、電壓等感應器等。

在過去5年來，許多新類型的攻擊技術被公開發表，如錯誤攻擊技術、時間攻擊技術、簡單電力攻擊技術、差異電力攻擊技術與電磁幅射攻擊技術，這些攻擊技術是靠容易以低價設備取得的資訊，如電力消耗、執行時間、故障時的輸出與輸入行為、輻射、電力尖峰情形等資訊攻擊

智慧卡。這些攻擊技術不但比傳統的攻擊技術優良，而且只要些許的資訊就可破解密碼系統。

於1995年 9月 29日，Paul Kocher首次描述密碼系統的執行時間特性與密秘金鑰是有關連性的觀念。並進一步在1996年提出攻擊者可如何分析這些時間特性以推導出密秘金鑰。自Kocher發表時間攻擊文獻後，人們才開始注意目前正在使用的產品與協定(如 SSL)，存在被攻擊的危險。

密碼演算法的執行時間常會因輸入資料的不同而有些許差異，這主要是因為密碼演算法會依不同的輸入資料執行不同路徑所致，這也就使得攻擊者可藉由分析收集到的執行時間資訊得以推演出密秘金鑰。

RSA密碼系統的基本運算為指數與模數運算、指模運算，且是被用來對訊息做加密或簽章動作。一般實作RSA密碼系統是使用如圖一計算 $M^e \bmod n$的指模演算法。圖一的指模演算法是一個從左至右、平方與乘法的演算法，其中M為需要加密的訊息，n為RSA系統的模數，指數 e為RSA系統的密秘金鑰，假設密秘金鑰 e有 w位元 ($e_{w-1} \cdots e_2 e_1 e_0$)，其演算法輸出 S是計算 $S = M^e \bmod n$的結果。

第一圖的演算法中，有一個重要現象值得注意，在該演算法中第 4 行的條件指令會影響執行路徑。在迴圈的

五、發明說明 (3)

第 k個圈次裡，假如密秘金鑰的第 k個位元 e_k為 1，則會執行第 3行與第 5行程式。類似的，假如密秘金鑰的第 k個位元 e_k為 0，則僅會執行第 3行程式。也就是說該演算法中第 4行條件指令的成立與否，與密秘金鑰的值有關。因此，整個計算 $M^e \bmod n$的計算時間，即演算法執行時間，會因密秘金鑰的值的不同而有所不同。

假如攻擊者依據第一圖的演算法，能夠觀察到該演算法不同圈次的執行時間，且能比較該演算法不同圈次執行時間的差異，該名攻擊者就可能推演出指數 e。當 RSA密碼系統應用於簽章時，上述的技術就可能推演出簽章者的密秘金鑰。此種攻擊技術稱為時間攻擊技術。

接下來說明簡單電力攻擊技術，簡單電力攻擊技術是一種直接觀察密碼運算時的電力消耗圖技術。第二圖為第一圖演算法的部份電力消耗剖面圖，圖中的 9個針峰為為平方運算與乘法運算的開始點。由於乘法運算比平方運算多一個載入的動作，所表現出來就是針峰比較寬。根據第一圖的 RSA演算法，若密秘金鑰的第 k個位元為 0則執行平方運算，反之則執行平方與乘法運算。依據平方較窄針峰、乘法較寬針峰的推論，因此若密秘金鑰的第 k個位元為 0則會出現較窄針峰，而若密秘金鑰的第 k個位元為 1則會出現較窄針峰後緊隨較寬針峰的結論，根據第二圖的電力消耗圖可推論出密秘金鑰的 5個位元為：00111。

要實作簡單電力攻擊技術，首先必須執行密碼演算法，在執行時同步記錄其電力消耗。攻擊者若知道執行指令及其相對應電力消耗圖的所在位置，那麼攻擊者就可以很快的獲得有用的資訊。換言之，簡單電力攻擊技術是建立在特定指令及其相對應電力消耗圖的所在位置的基礎上，因此攻擊者需要清楚的知曉密碼演算法的實作。

從前面時間攻擊技術與簡單電力攻擊技術的說明，可得到一個簡單結論，攻擊技術導因於條件指令的存在。有了此簡單結論，從第一圖改良而來的第三圖演算法被提出以防止時間攻擊技術與簡單電力攻擊技術的攻擊。在第三圖的演算法中迴圈內已沒有條件指令存在，如此一來整個演算法的時間就與密秘金鑰無關，而能免除時間攻擊與簡單電力攻擊。

由於攻擊技術的日新月異，新的且比較實際的攻擊技術不斷被提出，因此以往可以防某些攻擊的對策，就不見得可應付新的攻擊技術。在"Sung-Ming Yen, Seung-Joo Kim, Seon-Gan Lim, and Sang-Jae Moon. A counter-measure against one physical cryptanalysis may benefit another attack."提出一種錯誤攻擊技術 C safe error;而"Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-

based cryptanalysis."提出另一種錯誤攻擊技術M safe error，下一節將看到第三圖的防止時間攻擊技術與簡單電力攻擊技術的演算法將不能抵抗上述所提的C safe error和M safe error錯誤攻擊。

再接下來說明C safe-error攻擊，所謂C safe-error攻擊是於算術與邏輯運算單元內製作任何可使其產生暫時性計算錯誤的攻擊，有時此種暫時性錯誤不影響模運算結果，而有時即會導致模運算結果也錯誤，若模運算結果有時不影響有時又有影響的情況出現，而導致洩漏密秘金鑰，此種攻擊技術稱之為C safe-error攻擊技術。例如第三圖的演算法，若 $e_k=0$，$S_b=(S_b \cdot S_2) \bmod n$ 運算是多餘的運算，也就是說假如算術與邏輯運算單元在做此運算時，發生暫時性錯誤而導致本運算計算錯誤，但並不會影響運算結果。因此假如攻擊者在第k圈次執行 $S_b=(S_b \cdot S_2) \bmod n$ 運算時製造任何影響算術與運算單元的計算錯誤，觀看此錯誤是否影響整個模運算邏輯法的結果，若指模演算法結果有誤即可推論出密秘金鑰的第k位元 e^k 是1，反之若指模演算法結果無誤即可推論出密秘金鑰的第k位元 e^k 是0。

接下來再說明M safe-error攻擊，設定指令 $Y=X \cdot Y$，被乘數為X與乘數為Y，乘後結果存回Y。由於在指模演算法中的被乘數與乘數皆是很大的數目（假設為1024位元）

，而一般的CPU或密碼加速器僅有32位元的乘法器，因此在計算X · Y運算時會分區塊計算，被乘數X乘上區塊（32位元）的乘數Y_0，X · Y_0，其中Y_0為乘數的最右邊32位元，其積存至暫時儲存體。接下來計算X · Y_1，其積與右移32位元暫時儲存體的結果相加，其和再存回暫時儲存體。依此類推，如同手算方式般計算X · Y運算。X · Y運算完成時其積是存在暫時儲存體，再把暫時儲存體的值存回Y。

整個X · Y的運算是X · Y_0、X · Y_1、X · Y_2……X · Y_3依次計算。當計算到X · Y_{10}時，前面10個區塊（Y_0、Y_1、……、Y_9）將不再用到，若此時製作使存放（Y_0、Y_1、……、Y_9）的暫存器或儲存體區段錯誤，導致其值改變。又由於Y = X · Y指令最後會把積存至Y，就會把前面的錯誤掩蓋過去，而不影響模運算結果。若把被乘數與乘數對調，即Y = Y · X，套用前述觀念，製造出暫時性錯誤於乘數X，很明顯的錯誤將繼續存在不被掩蓋而導致錯誤的積。

此種暫時性錯誤有時不影響模運算結果，而有時確會導致模運算結果也錯誤，就會導致洩漏密秘金鑰，此種攻擊技術稱之為M safe-error攻擊技術。

接下來，將說明為何第三圖之演算法不能防止M

五、發明說明 (7)

safe-error或 C safe-error 的錯誤攻擊。

參考第三圖的演算法，若 e_k =0時，明顯發現 $S_1=(S_2 \cdot S_1) \bmod n$運算是多餘的運算，假如 ALU在做此運算產生 C safe error，此暫時性錯誤並不影響運算結果。若 e_k =1時，需計算 $S_0=(S_2 \cdot S_0) \bmod n$，假如 ALU在做此運算產生 C safe error，此暫時性錯誤將會產生錯誤的 S_0，此錯誤的 S將使下一個圈次計算 $S_0=(S_0 \cdot S_0) \bmod n$時產生錯誤計算而導致錯誤的結果。因此假如攻擊者在第 k圈次執行 $S_1=(S_2 \cdot S_1) \bmod n$運算時製造 C safe error，然而觀看此錯誤是否影響整個模運算法的結果是否有誤，若指模演算法結果有誤即可推論出密秘金鑰的第 k位元 e_k是 1，反之若指模演算法結果無誤即可推論出密秘金鑰的第 k位元 e_k是 0。因此第三圖的演算法並不能抵抗 C safe error的攻擊。

再看第三圖的演算法，若 e_k =0時，需計算 $S_1=(S_2 \cdot S_1) \bmod n$，若此時產生 M safe error，此暫時性錯誤並不影響運算結果。若 e_k =1時，需計算 $S_0=(S_2 \cdot S_0) \bmod n$，若此時產生 M safe error，此暫時性錯誤將會產生錯誤的 S_0，此錯誤的 S_0將使下一個圈次計算 $S_0=(S_0 \cdot S_0) \bmod n$時產生錯誤計算而導致錯誤的結果。因此假如攻擊者在第 k圈次執行 $S_b=(S_2 \cdot S_b) \bmod n$運算時製造 M safe error於 S_1，然而觀看此錯誤是否影響整個模運算法的結果是否有

誤，若指模演算法結果有誤即可推論出密秘金鑰的第 k位元 e_k是 1，反之若指模演算法結果無誤即可推論出密秘金鑰的第 k位元 e_k是 0。因此第三圖的演算法並不能抵抗 M safe error的攻擊。

由上述可知，攻擊者取得秘密金鑰的資訊，主要是透過時間攻擊，簡單電力攻擊，和 M-safe error attack 和 C safe-error attack即所謂的錯誤攻擊。

三、【發明內容】

鑑於以上所述，本專利的主要目的為設計一指模演算法，其運算時間不因輸入之指數而異，用以防止時間攻擊與簡單電力攻擊。

本發明的另一個目的為提供一指模演算法，使演算法迴圈內沒有多餘的運算，也就是說，不論攻擊者在迴圈所對應的秘密金鑰位元為 0或 1時產生錯誤攻擊，演算法所輸出的結果皆為錯誤。如此攻擊者即無法依照暫時性錯誤來推算秘密金鑰所對應的位元，以致於秘密金鑰不至於外洩，並進而達到防止 C safe error的攻擊。

本發明再一個目的為提供一指模演算法沒有被乘數乘以乘數存回不固定位址之暫時儲存體的操作，用以防止 M safe-error的攻擊。

承上所述，有別於傳統無法防止時間攻擊和簡單電力攻擊的演算法，本發明提供了一指模演算法，其中不包含條件指令以防止時間攻擊及簡單電力攻擊；並且演算法中沒有多餘的運算，使得攻擊者無法由藉由產生C safe error來得知秘密金鑰的資訊，且演算法中不包含被乘數乘以乘數存回不固定位址之暫時儲存體的操作，使得M safe error的攻擊不會對秘密金鑰產生威脅。並將指模運算中部分之被乘數乘以乘數再平方需兩次乘法運算的動作用被乘數先作平方再乘以乘數兩次需要三次乘法運算之方式來取代，以達到使指數運算所須的乘法次數不會因指數的不同而改變，進而達到防止時間攻擊。

本發明提供一保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法或裝置或為電腦可讀取媒體，其執行步驟包含：1. 取得需透過一公開金鑰系統加密之一信息；2. 取得相對於該公開金鑰系統之一模數與一秘密金鑰，其中之秘密金鑰由至少一個位元所組成；3. 將一第一值設為1，將該信息指定到一第二值；4. 將該秘密金鑰所對應之位元，由一最高位元至一最低位元依序進行一指模演算法，該指模演算法步驟包含：a. 將一運算位元通過一反相器並將輸出指定為一第三值，將此運算位元之次一位元指定為一第四值；b. 若第三值為0，則先計算第一值平方後對模數做模運算並將結果存於第一值，若第三值為1，則

先計算第一值乘以第二值後對模數做模運算並將結果存於第一值；c. 若第四值為0，則計算該第一值平方後對模數做模運算並將結果存於第一值，若第四值為1，則先計算第一值乘以第二值後對模數做模運算並將結果存於第一值；5. 將運算位元之次一位元存至運算位元，並對運算位元進行指模運算法之步驟，直到運算位元為最低位元；6. 將最終之第一值結果儲存並輸出。

四、【實施方式】

本發明的一些實施例會詳細描述如下。然而，除了詳細描述外，本發明還可以廣泛地在其他的實施例施行，且本發明的範圍不受限定，其以之後的專利範圍為準。

在習知技術中，第一圖所提供的演算法為一廣泛使用運算 R=M^e mod n的已知技術。其中 "^" 代表次方．基底M為傳送的信息，e為指數，而n為模數。如前所述，第一圖所提供的演算法攻擊者可藉由演算法執行乘法 S=(S· M) mod n與否來判斷秘密金鑰所對應的第k位元為1或0。藉由由左至右的位元計算，攻擊者可取得所有秘密金鑰的位元。

以下將詳細說明本發明所用以保護公開密碼系統的演算法，此演算法不但可以防止前述的時間攻擊，簡單電力攻擊，更可進一步的防制C safe-error攻擊和M safe-

error攻擊。不止是RSA公開金鑰系統，所有基於離散對數之系統皆可應用於本發明之演算法。

為防止錯誤攻擊技術的攻擊，本文提出如第四圖的指模演算法。當 e_k =0時執行 $R_0 \leftarrow (R_0 \cdot R_1) \bmod n$ 與 $R_0 \leftarrow (R_0 \cdot R_c) \bmod n$，當 e_k =1時執行 $R_0 \leftarrow (R_0 \cdot R_0) \bmod n$ 與 $R_0 \leftarrow (R_0 \cdot R_c) \bmod n$。第四圖的演算法沒有條件指令所以能防時間攻擊與電力攻擊。在迴圈內沒有多餘的運算，所以能防止C safe error的攻擊。沒有被乘數乘以乘數存回乘數的運算，所以能防止M safe error的攻擊。

本發明提供一演算法如第五圖，先取得需透過一公開金鑰系統加密之一信息M;並取得相對於該公開金鑰系統之一模數n與一秘密金鑰e，其中秘密金鑰為 $(e_{w-1}, e_{w-2}, e_{w-3} \cdots e_1, e_0)$，將S設為1，將信息M指定到 S_1，並將 e_{-1}之初值設為1(100)並進行下列步驟：

首先由最高位元 (e_{w-1})至最低位元 (e_0) 分別進行一指模演算法，並將k=w-1(110)，該指模演算法步驟包含：1將運算位元 e_k通過一反相器並將輸出指定為一b值，將該運算位元之次一位元 e_{k-1}指定到c值(120)；2.接著執行計算 $S_0 = (S_0 \cdot S_b) \bmod n$與 $S_0 = (S_0 \cdot S_c) \bmod n$(130)；3.接著將執行k=k-1(140)；4.重複第1步驟至第3步驟直到完成最終k=0的迴圈；5.最後將最終S儲存並輸出(150

)。

接下來將以舉例說明此演算法的正確性。第四圖的演算法假設密秘金鑰的最高位元 e_{w-1}為 1，當 e_k =0時執行 S_0 = (S_0· S_1)mod n與 S_0 =(S_0· S_c)mod n，當 e_k =1時執行 S_0 = (S_0· S_0)mod n與 S_0 = (S_0· S_c)mod n，其中 c= e_{k-1}。假設密秘金鑰為 10001100，追蹤第一圖與第四圖的演算法的內容如第六圖所示。為了在迴圈的每個圈次皆執行 2個運算，第四圖之演算法把 e_k =0 時要執行的 S=(S· S)mod n運算提前至前一個圈次計算，由於提前一個圈次計算使得該圈次要計算的 S=(S· M)mod n運算必需延後運算，若執行順序的 1. S=(S· M) mod n 2. S=(S· S)mod n被改變成 S=(S· S)mod n先計算，從數學推理上接下來的 S=(S· M)mod n必須算 2 次才能得相同的結果，因此若 S=(S· S)mod n先計算，則執行順序為 1. S=(S· S) mod n 2. S=(S· M) mod n 3. S=(S· M)mod n。

一開始第 7個圈次 e_7=1，執行 S_0= (S_0· S_0)mod n 與 S_0 =(S_0· S_c)mod n，其中 c= e_6。使得第 7個圈次執行 S_0 =(S_0· S_0)mod n 與 S_0 =(S_0· S_0)mod n運算，其中第 1個 S_0=(S_0· S_0)mod n運算是原來的運算，由於 e_6=0，S_0 = (S_0· S_0)mod n提前計算，使得第 2個 S_0 =(S_0· S_0)mod n則是屬於提前計算。

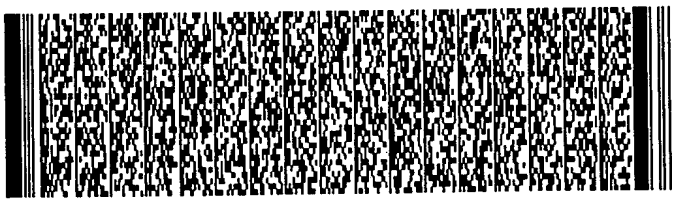
五、發明說明 (13)

下一個圈次第 6個圈次 $e_6=0$，執行 $S_0 = (S_0 \cdot S_1) \bmod n$ 與 $S_0 = (S_0 \cdot S_c) \bmod n$，其中 $c = e_5$。使得第 6個圈次執行 $S_0 = (S_0 \cdot S_1) \bmod n$ 與 $S_0 = (S_0 \cdot S_0) \bmod n$運算，其中 $S_0 = (S_0 \cdot S_1) \bmod n$運算是原來的運算，由於 $e_5=0$，$S_0 = (S_0 \cdot S_0) \bmod n$提前計算，使得第 2個 $S_0 = (S_0 \cdot S_0) \bmod n$則是屬於提前計算。本來此圈次要計算 2次 $S_0 = (S_0 \cdot S_1) \bmod n$，僅能計算一次，另外一次要延至下一個圈次。由於有 $S_0 = (S_0 \cdot S_0) \bmod n$計算，因此延至下一個圈次的 $S_0 = (S_0 \cdot S_1) \bmod n$運算，在下一個圈次時也要計算 2次。

下一個圈次第 5個圈次 $e_5=0$，與第 6個圈次情況相同不再討論。

下一個圈次第 4個圈次 $e_4=0$，執行 $S_0 = (S_0 \cdot S_1) \bmod n$ 與 $S_0 = (S_0 \cdot S_c) \bmod n$，其中 $c = e_3$。由於 $e_3=1$，使得第 4個圈次執行 $S_0 = (S_0 \cdot S_1) \bmod n$ 與 $S_0 = (S_0 \cdot S_1) \bmod n$。第 5個圈次時留下 $S_0 = (S_0 \cdot S_1) \bmod n$運算要第 4個圈次做，前面說過留下的 $S_0 = (S_0 \cdot S_1) \bmod n$必需做 2次，此一圈次剛好補完前一個圈次所留下的運算。

下一個圈次第 3個圈次 $e_3=1$，執行 $S_0 = (S_0 \cdot S_0) \bmod n$ 與 $S_0 = (S_0 \cdot S_c) \bmod n$，其中 $c = e_2$。由於 $e_2=1$，使得第 3個圈次執行 $S_0 = (S_0 \cdot S_0) \bmod n$ 與 $S_0 = (S_0 \cdot S_1) \bmod n$運算，這與演算法 1相同不在討論。



第2個圈次的情況與第7個圈次相同，第1個圈次的情況與第6個圈次相同，因此不再討論。

然後談最後一個圈次，第0個圈次 $e_0=0$，執行 $S_0=(S_0 \cdot S_1) \bmod n$ 與 $S_0=(S_0 \cdot S_c) \bmod n$，其中 $c=e_{-1}$。在此演算法中假設 $e_{-1}=1$，因此情況如第4個圈次一般，執行 $S_0=(S_0 \cdot S_1) \bmod n$ 與 $S_0=(S_0 \cdot S_1) \bmod n$，剛好補完前一個圈次所留下的運算。

由上述說明可知，本發明所提供的演算法，其中不包含條件指令以防止時間攻擊及簡單電力攻擊；並且演算法中沒有多餘的運算，使得攻擊者無法藉由產生C safe error來得知秘密金鑰的資訊，且演算法中不包含被乘數乘以乘數存回不固定位址之暫時儲存體的操作，使得M safe error的攻擊不會對秘密金鑰產生威脅。本發明所提供之演算法提供一計算 $R=M^e \bmod n$ 的方法，其所關連之電路裝置及程式皆可透過簡單的習知技術來獲得，故不在此說明書內加以描述。

以上所述僅為本發明之較佳實施例而已，此實施例僅係用來說明而非用以限定本發明之申請專利範圍。在不脫離本發明之實質內容的範疇內仍可予以變化而加以實施，此等變化應仍屬本發明之範圍。因此，本發明之範疇係由

以下之申請專利範圍所界定。

五、【圖式簡單說明】

第一圖為計算 Me mod n的指模演算法；

第二圖為演算法的部分電力消耗剖面圖；

第三圖為防止時間攻擊技術與簡單電力攻擊技術的 Me mod n演算法；

第四圖表示能防止錯誤攻擊的模演算法；

第五圖為第四圖演算法之流程圖；及

第六圖為在秘密金鑰為 10001100時第一圖與第四圖的演算法的追蹤內容。

主要部分之代表符號：

100-150 流程步驟。

六、申請專利範圍

1. 一種保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法，其步驟包含：

取得需透過一公開金鑰系統加密之一信息；

取得相對於該公開金鑰系統之一模數與一秘密金鑰，其中，該秘密金鑰由至少一個位元所組成；

將一第一值設為1，將該信息指定到一第二值；

將該秘密金鑰所對應之該些位元，由一最高位元至一最低位元依序進行一指模演算法，該指模演算法步驟包含：

將一運算位元通過一反相器並將輸出指定為一第三值，將該運算位元之次一位元指定為一第四值；

若該第三值為0，則先計算該第一值平方後對該模數做模運算並將結果存於該第一值，若該第三值為1，則先計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；以及

若該第四值為0，則計算該第一值平方後對該模數做模運算並將結果存於該第一值，若該第四值為1，則先計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；

將該運算位元之次一位元存至該運算位元，並對該運算位元進行該指模運算法之步驟，直到該運算位元為該最低位元；以及

將最終該第一值結果儲存並輸出。

六、申請專利範圍

2. 如申請專利範圍第1項保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法，其中若該運算位元為該秘密金鑰之該最低位元，則該第四值設為1。

3. 如申請專利範圍第2項保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法，其中該運算位元為該秘密金鑰中之該些位元之一，其可為二位元制中之0或1。

4. 如申請專利範圍第1項保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法，其中該反相器作用為將輸入位元0反相輸出為1，輸入位元1反相輸出為0。

5. 一種保護公開金鑰系統以防止時間、電力與錯誤攻擊之裝置，包含：

一裝置用以取得欲透過一公開金鑰系統加密之一信息；

一裝置用以取得相對於該公開金鑰系統之一模數與一秘密金鑰，其中，該秘密金鑰由至少一個位元所組成；

一裝置用以將一第一值設為1，將該信息指定到一第二值；

一裝置用以將該秘密金鑰所對應之該些位元，由該些位元之一最高位元至一最低位元分別進行一指模演算法，該指模演算法裝置包含：

一裝置將一運算位元通過一反相器並將輸出指定

為一第三值，將該運算位元之次一位元指定為一第四值；

一裝置用以判斷第三值為0或1；

一裝置，當該第三值為0，用以計算該第一值平方後對該模數做模運算並將結果存於該第一值，當該第三值為1，計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；

一裝置用以判斷若該第四值為0或1；以及

一裝置當該第四值為0，用以計算該第一值平方後對該模數做模運算並將結果存於該第一值，當該第四值為1，計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；

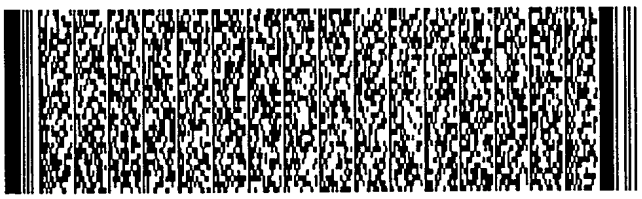
一裝置用以將該運算位元之次一位元存回該運算位元，並對該運算位元進行該指模演算法之步驟；

一裝置用以判斷該運算位元是否為該最低位元；以及

一裝置將該第一值結果儲存並輸出。

6. 如申請專利範圍第5項保護公開金鑰系統以防止時間、電力與錯誤攻擊之方法，其中該運算位元為二進位制中之0或1。

7. 如申請專利範圍第5項保護公開金鑰系統以防止時間、電力與錯誤攻擊之裝置，其中該反相器作用為將輸入位元0反相輸出為1，輸入位元1反相輸出為0。



六、申請專利範圍

8. 一種保護公開金鑰系統以防止時間、電力與錯誤攻擊之電腦可讀取媒體，其程式碼執行步驟包含：

取得需透過一公開金鑰系統欲加密之一信息；

取得相對於該公開金鑰系統之一模數與一秘密金鑰，其中，該秘密金鑰由至少一位元所組成；

將一第一值設為1，將該信息指定到一第二值；

依序對該秘密金鑰所對應之一最高位元至一最低位元，針對該位元執行多個步驟：

將該位元通過一反相器並將輸出指定為一第三值，將該位元之次一位元指定為一第四值；

若該第三值為0，則先計算該第一值平方後對該模數做模運算並將結果存於該第一值，若該第三值為1，則先計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；以及

若該第四值為0，則計算該第一值平方後對該模數做模運算並將結果存於該第一值，若該第四值為1，則先計算該第一值乘以該第二值後對該模數做模運算並將結果存於該第一值；

將該位元之次一位元存回該位元，並針對該位元執行該些步驟，直到該位元為該最低位元；以及

將最終該第一值結果儲存並輸出。

9. 如申請專利範圍第8項保護公開金鑰系統以防止時間、電力與錯誤攻擊之電腦可讀取媒體，其中若該運算位元為

該秘密金鑰之該最低位元，則該第四值設為1。

10. 如申請專利範圍第9項保護公開金鑰系統以防止時間、電力與錯誤攻擊之電腦可讀取媒體，其中該反相器作用為將輸入位元0反相輸出為1，輸入位元1反相輸出為0。

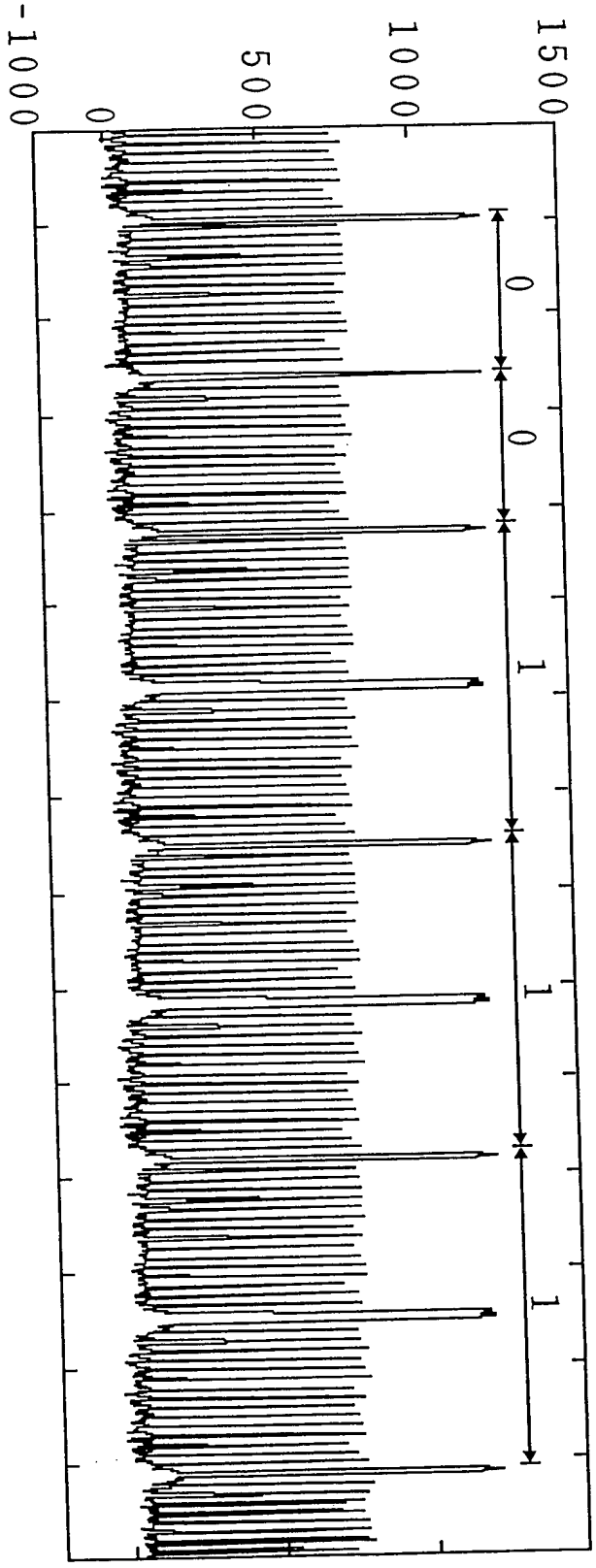
11. 如申請專利範圍第8項保護公開金鑰系統以防止時間、電力與錯誤攻擊之電腦可讀取媒體，其中該位元為該秘密金鑰中該些位元之一，其可為0或1。

輸入: M, n, e = ($e_{w-1}\cdots e_2e_1e_0$)
輸出: $S = M^e \bmod n$

```
1   Let S = 1
2   FOR k = w-1 downto 0
3         S = (S•S) mod n
4         IF (ek is 1 ) THEN
5               S = (S•M) mod n
6         ENDIF
6   ENDFOR
7   RETURN S
```

第一圖

圖式



第二圖

輸入: M, n, e = ($e_{w-1}\cdots e_2e_1e_0$)
輸出: $S = M^e \bmod n$

```
1   Let S0 = 1; S2 = M
2   FOR k = w-1 downto 0
3         b = ~ek
4         S0 = (S0•S0) mod n
5         Sb = (S2•Sb) mod n
6    ENDFOR
7   RETURN S0
```

第三圖

輸入: M, n, e = ($e_{w-1}\cdots e_2e_1e_0$)

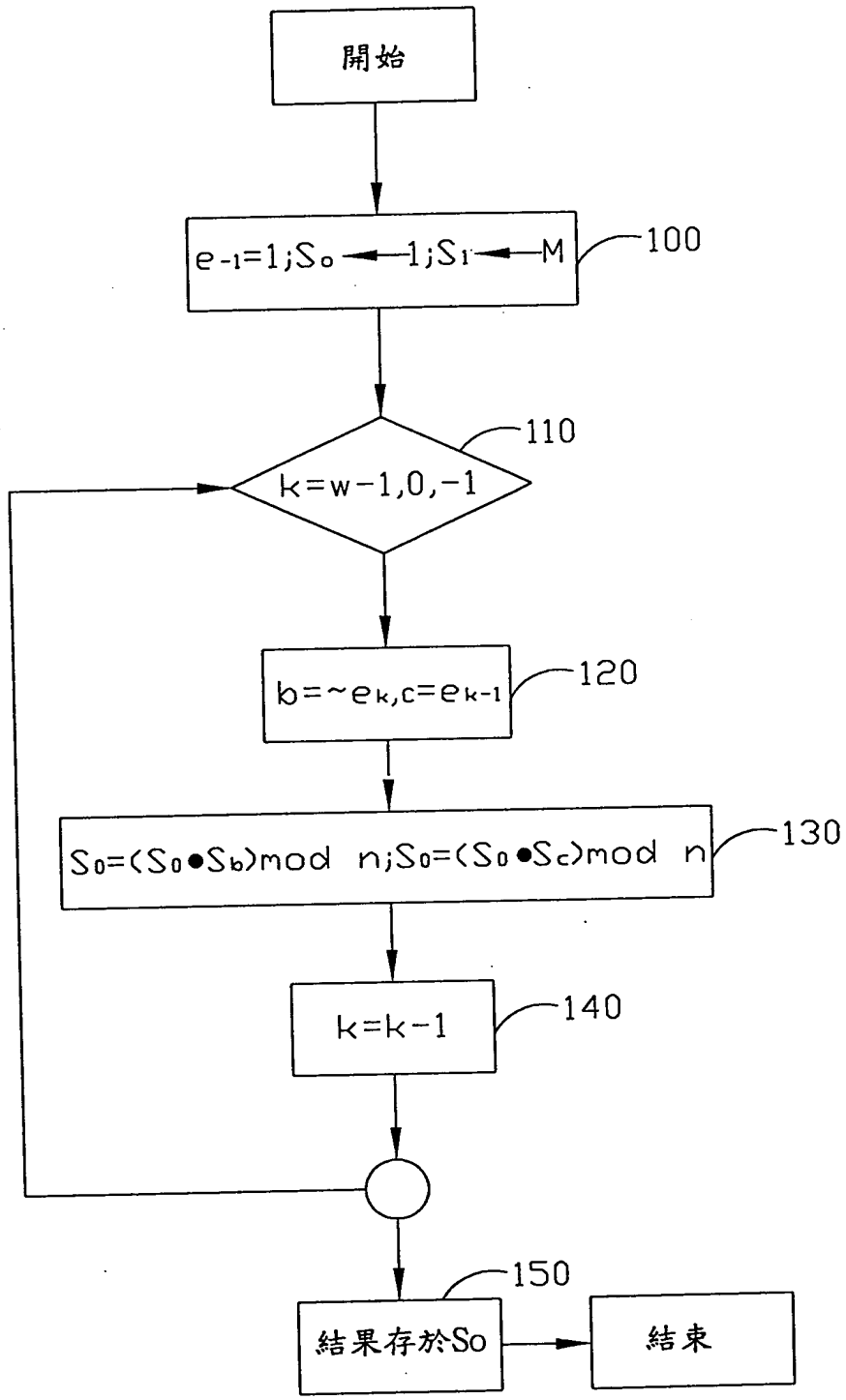
輸出: $S_o = M^e \bmod n$

演算法：assume $e_{w-1}=1$

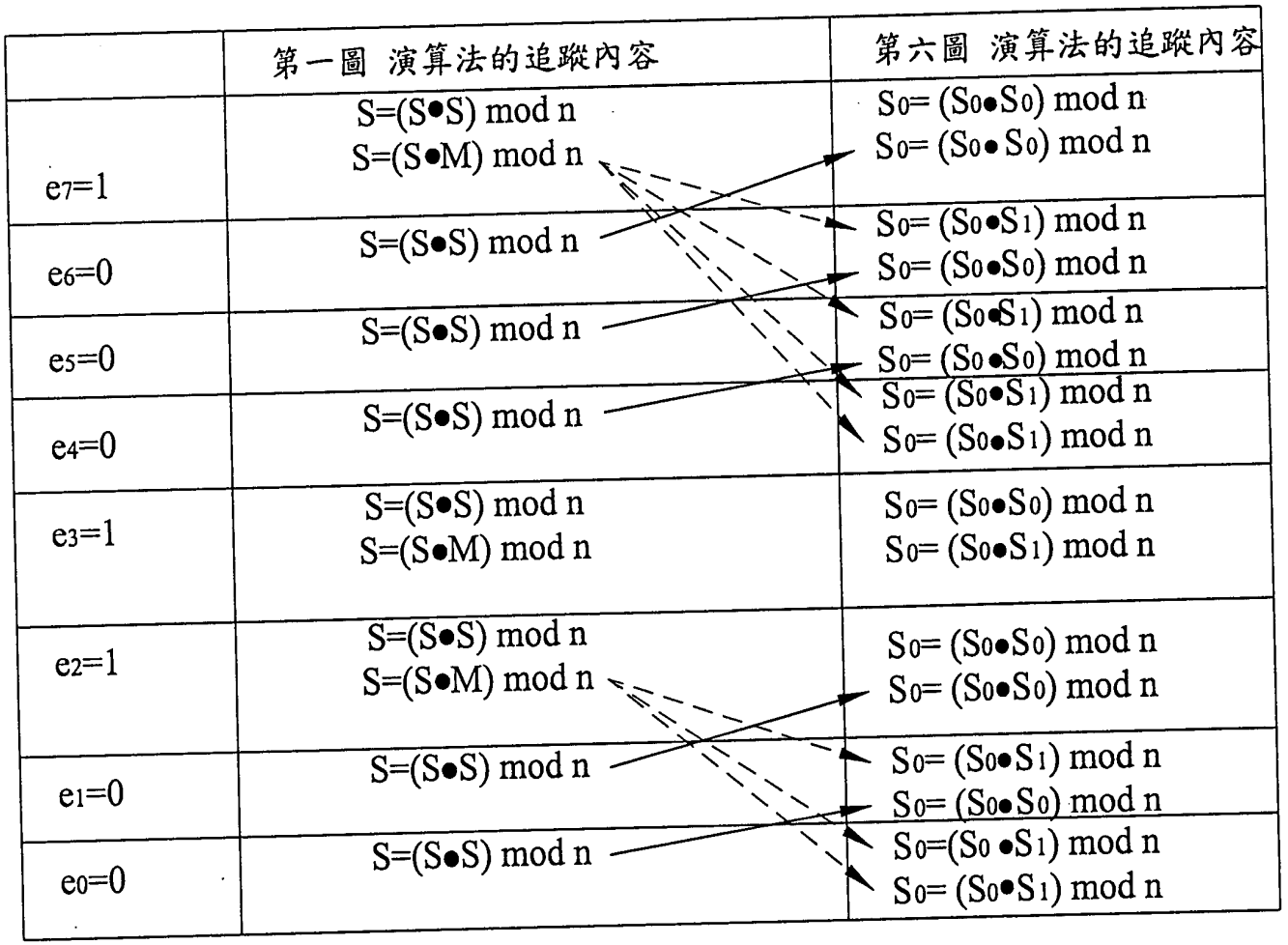
```
1.  e-1=1
2.  S0 = 1 ;  S1 = M
3.  FOR k = w-1 downto 0 DO
4.        b = ~ek ; c = ek-1
5.        S0 = (S0 •Sb) mod n; S0 = (S0 •Sc) mod n
    ENDFOR
6.  RETURN S0
```

第四圖

圖式


開始
e-1=1;So←1;S1←M
100
110
k=w-1,0,-1
b=~ek,c=ek-1
120
So=(So●Sb)mod n;So=(So●Sc)mod n
130
k=k-1
140
150
結果存於So
結束

第五圖

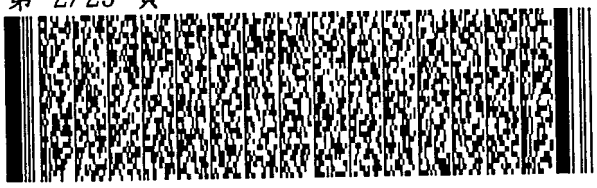

第一圖 演算法的追蹤內容
第六圖 演算法的追蹤內容
e7=1
S=(S•S) mod n
S=(S•M) mod n
S0= (S0•S0) mod n
S0= (S0• S0) mod n
e6=0
S=(S•S) mod n
S0= (S0•S1) mod n
S0= (S0•S0) mod n
e5=0
S=(S•S) mod n
S0= (S0•S1) mod n
S0= (S0•S0) mod n
e4=0
S=(S•S) mod n
S0= (S0•S1) mod n
S0= (S0•S1) mod n
e3=1
S=(S•S) mod n
S=(S•M) mod n
S0= (S0•S0) mod n
S0= (S0•S1) mod n
e2=1
S=(S•S) mod n
S=(S•M) mod n
S0= (S0•S0) mod n
S0= (S0•S0) mod n
e1=0
S=(S•S) mod n
S0= (S0•S1) mod n
S0= (S0•S0) mod n
e0=0
S=(S•S) mod n
S0=(S0 •S1) mod n
S0= (S0•S1) mod n

第六圖

第 1/25 頁

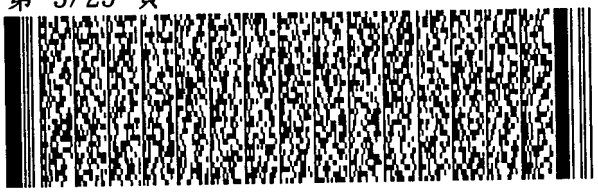
第 2/25 頁

第 2/25 頁

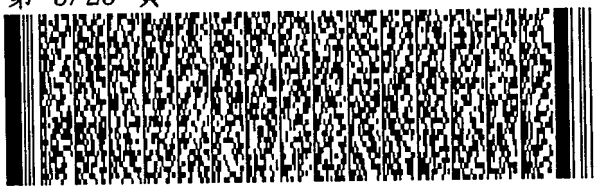


第 3/25 頁

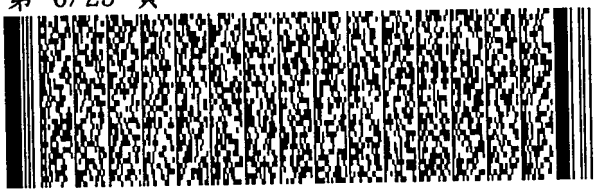
第 5/25 頁



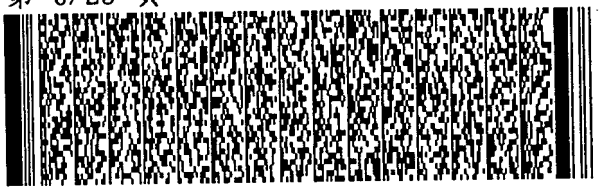
第 5/25 頁



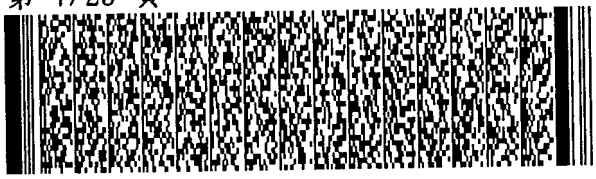
第 6/25 頁



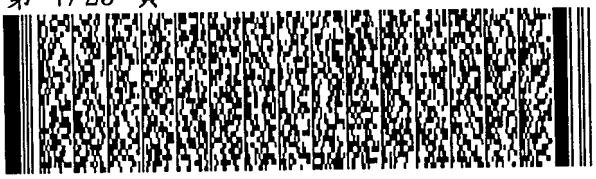
第 6/25 頁



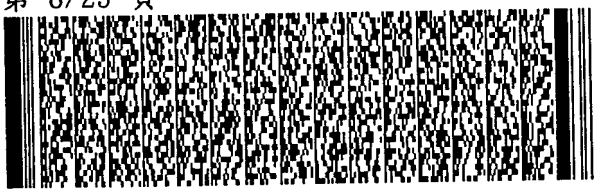
第 7/25 頁



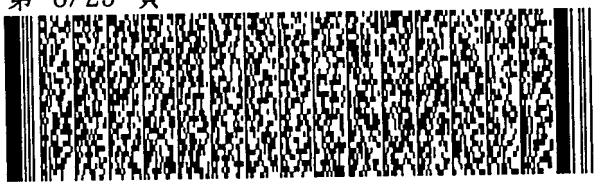
第 7/25 頁



第 8/25 頁

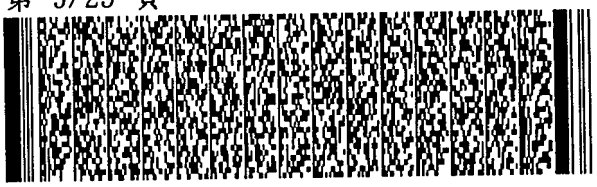


第 8/25 頁



第 9/25 頁

第 9/25 頁

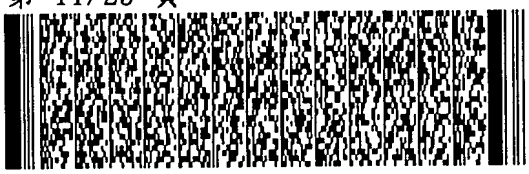


第 10/25 頁

第 10/25 頁

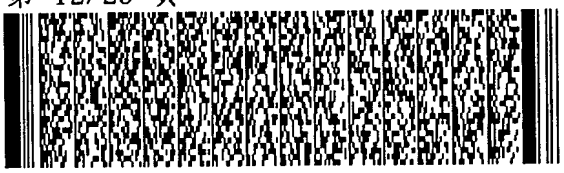
第 11/25 頁

第 11/25 頁

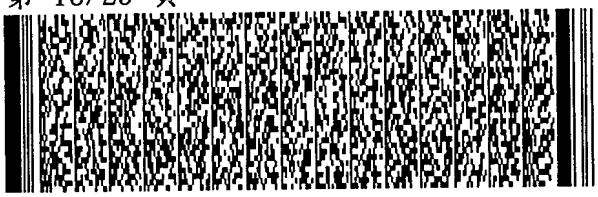


第 12/25 頁

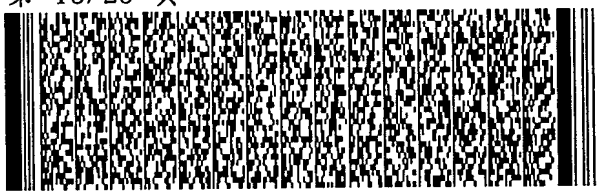
第 12/25 頁



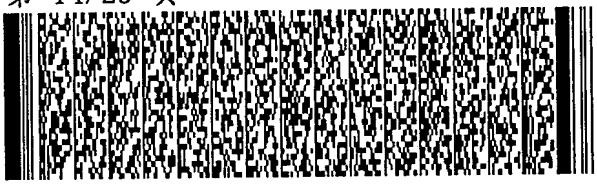
第 13/25 頁



第 13/25 頁

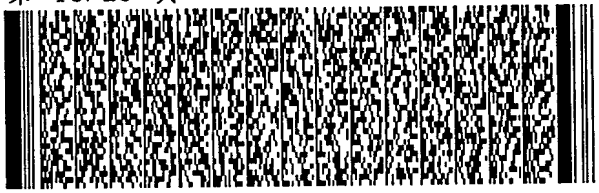


第 14/25 頁

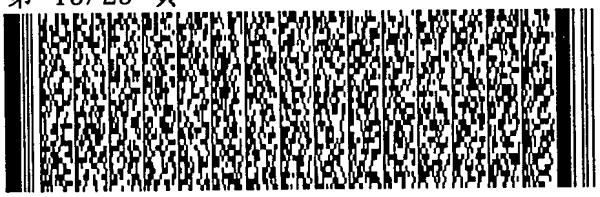


第 14/25 頁

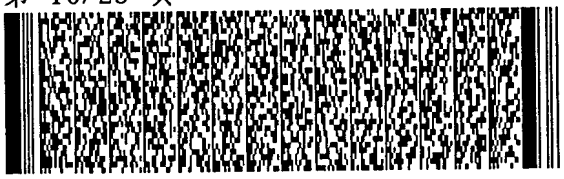
第 15/25 頁



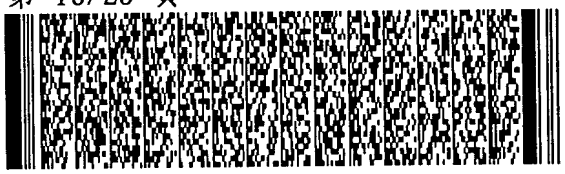
第 15/25 頁



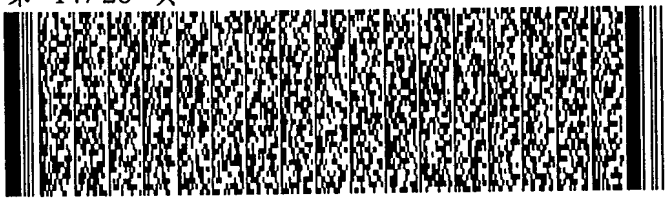
第 16/25 頁



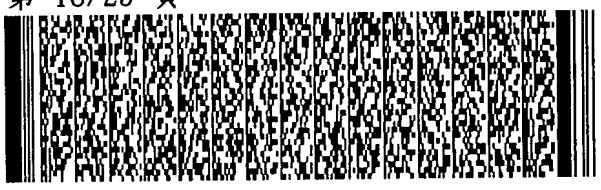
第 16/25 頁



第 17/25 頁



第 18/25 頁

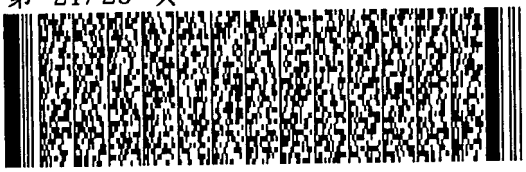


第 18/25 頁

第 19/25 頁

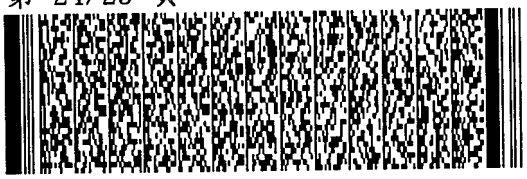
第 20/25 頁

第 21/25 頁

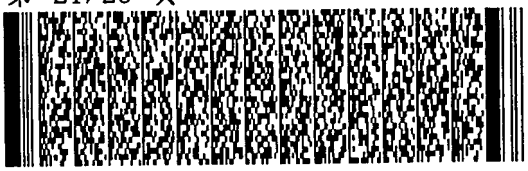


第 23/25 頁

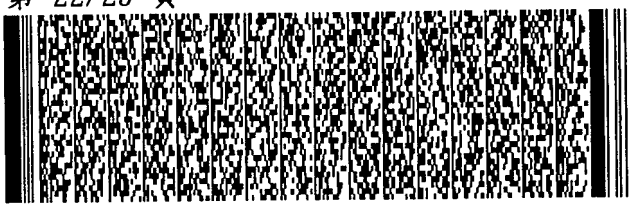
第 24/25 頁



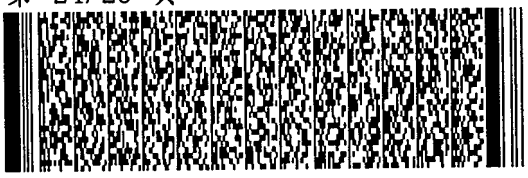
第 21/25 頁



第 22/25 頁



第 24/25 頁



第 25/25 頁